The Class Constructor

A class constructor is a specialized member function of a class. The distinctive property of a constructor is that it is always executed whenever we create new objects of that class.
A **constructor has exactly the same name as the class** and it does not have any return type, not even void. Constructors can be very useful for setting initial values for certain member variables.
In C++, if the computer hangs before the constructor can fully execute, the object will not be created since the constructor is responsible for initializing the object, and if it doesn't complete its execution, the object remains in an undefined state.

In **default constructor**, if we set the default length to 0, objects will not have garbage values as lengths.

Parameterized Constructor – takes in arguments.

Copy Constructor – copies objects into other objects. By default, we have a copy constructor without explicitly declaring and defining it. It copies all attributes. However, if we declare a custom one, we have a choice to define the copying procedure.

We **can't call a copy constructor by value** because in case of copy by value (and not as a reference), the object will be copied and it will call the copy constructor member function again and ultimately infinite recursive calls will be made to the same function.

If we write a normal constructor and don't write a copy constructor, **the compiler has a default copy** constructor to copy all the attributes.

Destructor is an instance member function that is invoked automatically whenever an object is going to be destroyed and it is the last function that is called before an object is destroyed.

Destructors overloading is not possible because the destructor is the only way to destroy the object created, and since the object has only one instance, it can be destroyed only once.